Docket No. RSW00-0010 ***PATENT***

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Attwood et al.**	§	
	§	Group Art Unit: **2134**
Serial No. **09/503,608**	§	
	§	Examiner: **Ellen C. Tran**
Filed: **February 11, 2000**	§	
	§	
For: **Technique of Defending Against Network Flooding Attacks Using a Connectionless Protocol**	§ §	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

36736
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on January 25, 2007.

No fees are believed to be required. If, however, any fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0461. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0461.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-6 and 9-14.

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: 7 and 8.
2. Claims withdrawn from consideration but not canceled: None.
3. Claims pending: 1-6 and 9-14.
4. Claims allowed: None.
5. Claims rejected: 1-6 and 9-14.
6. Claims objected to: None.

C. CLAIMS ON APPEAL

The Claims on appeal are: 1-6 and 9-14.

STATUS OF AMENDMENTS

No amendments were submitted after the Final Office Action of October 20, 2006.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to a method of preventing a flooding attack on a network server in which a large number of connectionless datagrams are received for queuing to a port on the network server (Specification, p. 1, ll. 7-10; p. 3, ll. 3-10; p. 12, ll. 5-7). The method includes determining, in response to the arrival of a connectionless datagram from a host for a port on the network server, if the number of connectionless datagrams already queued to the port from the host exceeds a prescribed threshold (Specification, p. 3, l. 11 – p. 4, l. 1; p. 4, ll. 7-13; p. 12, ll. 7-10); discarding the datagram, if the number of connectionless datagram already queued to the port from the host exceeds the prescribed threshold (Specification, p. 3, ll. 14-16; p. 4, ll. 1-4; p. 6, ll. 18-21; p. 12, ll. 10-11), and queuing the connectionless datagram to a queue slot of the port, if the number of connectionless datagram already queued to the port from the host does not exceed the prescribed threshold (Specification, p. 4, ll. 1-5; p. 6, ll. 21-26).

B. CLAIM 3 - INDEPENDENT

The subject matter of claim 3 is directed to an apparatus for preventing a flooding attack on a network server in which a large number of datagrams are received for queuing to a port on the server (Specification, p. 1, ll. 7-10; p. 3, ll. 3-10; p. 7, ll. 1-6; p. 12, ll. 5-7). The apparatus includes means for determining, in response to a datagram from a host for the port on the network server, if the number of datagrams queued on the port by the host exceeds a prescribed threshold (Specification, p. 3, l. 11 – p. 4, l. 1; p. 4, ll. 7-13; p. 7, ll. 1-6; p. 12, ll. 7-10), means for discarding the datagram, if the number of datagrams queued on the port by the host exceeds the prescribed threshold (Specification, p. 3, ll. 14-16; p. 4, ll. 1-4; p. 6, ll. 18-21; p. 7, ll. 1-6; p. 12, ll. 10-11), and means for queuing the datagram to a queue slot of the port, if the number of datagrams queued on the port by the host does not exceed the prescribed threshold (Specification, p. 4, ll. 1-5; p. 6, ll. 21-26; p. 7, ll. 1-6).

C. CLAIM 4 - DEPENDENT

Claim 4 is directed to the apparatus of claim 3, further including means for calculating the prescribed threshold by multiplying a percentage by a number of available queue slots for the port (Specification, p. 3, l. 18 – p. 4, l. 6; p. 5, ll. 2-9; p. 7, ll. 1-6).

D. CLAIM 5 - INDEPENDENT

The subject matter of claim 5 is directed to a storage media containing program code that is operable by a computer for preventing a flooding attack on a network server in which a large number of datagrams are received for queuing to a port on the network server (Specification, p. 1, ll. 7-10; p. 3, ll. 3-10; p. 7, ll. 1-6; p. 12, ll. 5-7). The program code includes instructions for causing the computer to execute the steps of determining, in response to receiving a datagram from a host for the port on the network server, if the number of datagrams already queued to the port from the host exceeds a prescribed threshold (Specification, p. 3, l. 11 – p. 4, l. 1; p. 4, ll. 7-13; p. 7, ll. 1-6; p. 12, ll. 7-10), discarding the datagram, if the number of datagrams already queued to the port from the host exceeds the prescribed threshold (Specification, p. 3, ll. 14-16; p. 4, ll. 1-4; p. 6, ll. 18-21; p. 7, ll. 1-6; p. 12, ll. 10-11), and queuing the datagram to a queue slot of the port, if the number of datagrams already queued to the port from the host does not exceed the prescribed threshold (Specification, p. 4, ll. 1-5; p. 6, ll. 21-26; p. 7, ll. 1-6).

E. CLAIM 12 - DEPENDENT

Claim 12 is directed to the apparatus of claim 3, further including means for configuring a maximum number of connectionless datagrams allowed to be queued at the port (Specification, p. 3, l. 18 – p. 4, l. 13; p. 6, ll. 14-26; p. 7, ll. 1-6).

F. CLAIM 13 - DEPENDENT

Claim 13 is directed to the apparatus of claim 12, wherein the means for configuring further comprises means for configuring a controlling percentage of available queue slots remaining for the port (Specification, p. 3, l. 18 – p. 4, l. 13; p. 6, ll. 14-26; p. 7, ll. 1-6).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to review on appeal are as follows:

A. GROUND OF REJECTION 1 (Claims 1, 3, 5, and 14)

Whether claims 1, 3, 5, and 14 fail to be anticipated under 35 U.S.C. §102 by *Schuba et al.*, Network Protection for Denial of Service Attacks, U.S. Patent 6,725,378, April 15, 1999 (hereinafter "*Schuba*").

B. GROUND OF REJECTION 2 (Claims 2, 4, 6, and 9-13)

Whether the Examiner failed to state a *prima facie* obviousness rejection against claims 2, 4, 6, and 9-13 under 35 U.S.C. §103(a) over *Schuba* in view of *Yavatkar et al.*, Method and System for Diagnosing Network Intrusion, U.S. Patent 6,735,702, August 31, 1999 (hereinafter "*Yavatkar*")

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1, 3, 5, and 14)

The first ground of rejection is whether claims 1, 3, 5, and 14 fail to be anticipated under 35 U.S.C. §102 by *Schuba et al.*, Network Protection for Denial of Service Attacks, U.S. Patent 6,725,378, April 15, 1999 (hereinafter "*Schuba*"). This rejection is in error and should be reversed.

A.1. Response to Rejection

Claim 1 is a representative claim in this grouping of claims. Claim 1 is as follows:

> 1. A method of preventing a flooding attack on a network server in which a large number of connectionless datagrams are received for queuing to a port on the network server, comprising:
>
> determining, in response to the arrival of a connectionless datagram from a host for a port on the network server, if the number of connectionless datagrams already queued to the port from the host exceeds a prescribed threshold;
>
> discarding the datagram, if the number of connectionless datagram already queued to the port from the host exceeds the prescribed threshold; and
>
> queuing the connectionless datagram to a queue slot of the port, if the number of connectionless datagram already queued to the port from the host does not exceed the prescribed threshold.

A prior art reference anticipates the claimed invention under 35 U.S.C. §102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re Bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed. Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). In this case each and every feature of the presently claimed invention is not identically shown in the cited reference, arranged as they are in the claims.

Regarding claim 1, the Examiner states that:

> As to independent claim 1, "A method of preventing a flooding attack on a network server" is taught in '378 col. 1, lines 55-60 "the present invention includes a unique defense for denial of service attacks";
>
> "in which a large number of connectionless datagrams are received for queuing to a port on the network server, comprising:" is shown in '378 col. 3, lines 16-33 "The Internet Protocol (IP) is the standard network layer protocol of the Internet that provides a connectionless, best effort packet delivery service. IP defines the basic unit of the data transfer used throughout an IP network, called a datagram. The deliver of datagrams is not guaranteed . . . Datagrams are routed towards their destination host" {"connectionless datagrams" same as "connectionless, best effort packet delivery service" / "network server" same as "destination host");
>
> "determining, in response to the arrival of a connectionless [sic] datagram from a host for a port on the network server" is disclosed in '378 col. 4, lines 52-54 "When a SYN packet arrives at a port on which a TCP server is listening";
>
> "if the number of connectionless; datagrams already queued to the port from the host exceeds a prescribed threshold discarding the datagram, if the number of connectionless datagrams already queued to the port from the host exceeds the prescribed threshold" is taught in '378 col. 4, lines 54-58 "There is a limit on the number of concurrent TCP connections that can be in a half-open connection state, called the SYN-RECVD state (i.e., SYN received). When the maximum number of half-open connections per port is reached, TCP discards all new incoming connections requests";
>
> "and queuing the connectionless datagram to a queue slot of the port, if the number of connectionless. datagrams already queued to the port from the host does not exceed the prescribed threshold" is taught in '378 col. 4, lines 59-67 "until it has either cleared or completed some of the half-open connections".

Final Office Action dated October 20, 2006, pp. 5-6.

Schuba does not anticipate claim 1 because *Schuba* does not disclose all of the features of claim 1. Specifically, *Schuba* fails to disclose the feature of connectionless datagrams already queued to the port as claimed in the determining, discarding, and queuing steps.

The Examiner cites the following portion of *Schuba* with respect to the determining, discarding, and queuing steps:

> *When a SYN packet arrives at a port on which a TCP server is listening, the above-mentioned data structures are allocated. There is a limit on the number of concurrent TCP connections that can be in a half-open connection state, called the SYN-RECVD state (i.e., SYN received). When the maximum number of half-open connections per port is reached, TCP*

discards all new incoming connection requests until it has either cleared or completed some of the half-open connections. Typically, several ports can be flooded in this manner, resulting in degraded service or worse. Moreover, it should be appreciated that without a limit on the number of half-open connections, a different denial of service attack would result in which an attacker could request so many connections that the target machine's memory is completely exhausted by allocating data structures for half-open TCP connections. Table II illustrates the half-open connection states that may be accommodated by various operating systems as follows:

TABLE II

Operating System	Backlog	Backlog + Grace
FreeBSD 2.1.5	n.a.	128
Linux 1.2.x	10	10
Solaris 2.4	5	n.a
Solaris 2.5.1	32	n.a.
SunOS 4.x	5	8
Windows NTs 3.51	6	6
Windows NTw 4.0	6	6

Schuba, col.4, l.52 - col.5, l.13 (emphasis to show portions cited by the Examiner).

The cited portion of *Schuba* teaches a method of defeating flooding attacks by limiting the number of half-open connections allowed at a given port. *Schuba* defines a half-open connection as a state in which the SYN datagram from a destination host has been received at a source host. As shown below, a half-open connection is not the same as a queue of datagrams, contrary to any assumptions or assertions the Examiner has made. Thus, *Schuba* does not teach the feature of connectionless datagrams already queued to the port as claimed in the determining, discarding, and queuing steps of claim 1. Accordingly, *Schuba* does not anticipate claim 1.

A half-open connection is not the same as a queue of datagrams. *Schuba* describes the process of establishing a transmission control protocol (TCP) connection in figure 1 of *Schuba*, which is reproduced below:

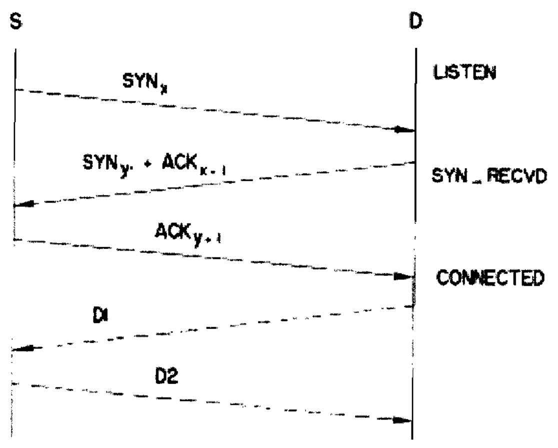

Fig. 1

Schuba, Figure 1.

Specifically, a three-way handshake occurs between a source host and a destination host. *Schuba* specifically states that before data can be transmitted between a source host and a destination host, a connection must be established between the two hosts. The source host sends a SYN datagram to the destination host. The destination host then sends a SYN+ACK datagram to the source host. At this point, the destination host is in the SYN_RECVD state. In this state, the destination host is waiting for an ACK datagram from the source host. The source host then sends an ACK datagram to the destination host, resulting in a connected state between the destination host and the source host. Thereafter, additional datagrams containing data are exchanged between the source host and the destination host. *Schuba* describes this process generally as follows:

> Referring to FIG. 1, a diagram is provided that illustrates the TCP packet sequence of a three-way handshake needed to establish a TCP connection. Before data can be transmitted between a source host S and a destination host D, TCP needs to establish a connection between source host S and destination host D. The connection establishment process is called the three-way handshake. The three-way handshake is established by exchanging certain TCP packet types between source host S and destination host D. The TCP packet types are distinguished by dedicated flag bits set in a TCP header code field and are listed in Table I as follows:

TABLE I

TCP Header Flag Bits	Abbreviation
SYNchronize	SYN
ACKnowledgement	ACK
ReSeT	RST

> It should be appreciated that, under appropriate conditions, more than one of the flag bits may be set in the same TCP packet.
>
> The first transmission in the three-way handshake is from source host S to destination host D in the form of a SYN packet (SYN flag bit set) while destination host D is in the LISTEN state. The second message, from destination host D to source host S, has both the SYN and ACK bit flags set (SYN+ACK) indicating that destination host D acknowledges the SYN packet and is continuing the handshake. At this point, destination host D is in the SYN_RECVD state. The third message, from source host S to destination host D has its ACK bit flag set, and is an indication to destination host D that both hosts S and D agree that a connection has been established, resulting in the CONNECTED state of destination host D. The third message may contain user payload data. Datagrams D1 and D2 represent data exchanges that take place after proper establishment-of the TCP connection.

Schuba, col.3, l.46 - col.4, l.16.

As stated in the cited portion, *Schuba* defines a half-open connection as a state in which the SYN datagram from a destination host has been received at a source host. Thus, the destination host is waiting for an ACK datagram from the source host. If the source host never sends an ACK datagram, then no full connection is established. However, the destination host still holds a half-open connection at a port in anticipation of receiving the ACK message. If the ACK message never arrives, then the port at the destination host can become unavailable. A malicious user can take advantage of this fact by causing the destination host to form enough half-open connections that no ports remain available to genuine users. *Schuba* specifically deals with this problem by limiting the number of half-open connections that are available at a given port at a destination host.

In contrast, the invention of claim 1 limits the <u>number of datagrams</u> that are allowed to <u>queue</u> at a given port. Specifically, claim 1 recites that an arriving datagram should be discarded <u>if the number of connectionless datagram already queued</u> to the port from the host exceeds the prescribed threshold. *Schuba* does not teach this claimed feature. *Schuba* teaches limiting the number of half-open connections by discarding new requests for connections. *Schuba* does not teach discarding datagrams to be assigned to a queue.

Similarly, *Schuba* does not teach the feature of determining, in response to the arrival of a connectionless datagram from a host for a port on the network server, <u>if the number of connectionless datagrams already queued</u> to the port from the host exceeds a prescribed threshold. Instead, *Schuba* counts the number of <u>half-open connections</u> at a port, which as described above, is entirely different than determining the number of connectionless datagrams already queued to a port.

Similarly, *Schuba* does not teach the feature of queuing the connectionless datagram to a queue slot of the port, if the number of connectionless datagrams already queued to the port from the host do not exceed the prescribed threshold. *Schuba* teaches establishing connections and limiting the number of half-open connections, not queuing datagrams at a port if the number of connectionless datagrams already queued to a port has not been exceeded.

In summary, *Schuba* does not teach the features of claim 1 because *Schuba* does not teach anything regarding determining the number of connectionless datagrams queued at a port or discarding or queuing the number of connectionless datagrams at a port. Instead, *Schuba* teaches determining the number of half-open connections at a port. The two features are entirely distinct for the reasons given above. Accordingly, *Schuba* does not anticipate claim 1.

A.2. Rebuttal to Examiner's Responses

In response to the facts established above regarding the feature of connectionless datagrams already queued to the port as claimed in the determining, discarding, and queuing steps, the Examiner asserts that:

> In response to applicant's argument beginning on page 8, "*Schuba* does not anticipate claim 1 because *Schuba* does not teach the claimed steps of determining, discarding, and queuing, as claimed ... As shown below, a half-open connection is not the same as a queue of datagrams, contrary to any assumptions or assertions the examiner has made. Thus, *Schuba* does not teach "determining, in response to the arrival of a connectionless datagram from a host for a port on the network server". The Office disagrees with argument, *Schuba* does show teach discarding, and queuing as claimed, in viewing the arguments and lengthy case history with this application examiner finds applicant is trying to differentiate the meaning of a connection attempt by using word such as connectionless or queuing the connectionless datagram. The protection against flooding attack as claimed is shown in *Schuba*.
>
> In response to applicant's argument beginning on page 9, "As shown below, a half-open connection is not the same as a queue of datagrams, *Schuba* describes the process of establishing a transmission protocol (TCP) connection in figure 1 of *Schuba* ... As stated above, *Schuba* defines a half-open connection as a state in which the SYN datagram from a destination host has been received at a source host ... In contrast, the invention of claim 1 limits the number of datagrams that are

> allowed to queue at a given port". The Office disagree with argument and notes that "half open-connections" are interpreted to be equivalent to "queuing the connectionless datagram".

Office Action dated May 2, 2006, pp. 2-3.

As shown above, *Schuba* does not teach the feature of connectionless datagrams already queued to the port as claimed in the determining, discarding, and queuing steps because a half-open connection is not the same as a queue of datagrams. *Schuba* instead teaches discarding incoming <u>connection requests</u> until the maximum number of <u>half-open connections</u> is reduced.

Several important differences exist between discarding additional connection requests, as in *Schuba*, and discarding the datagram, if the number of connectionless datagrams already queued to the port from the host exceeds the prescribed threshold, as recited in claim 1. For example, in *Schuba*, no queue <u>for the datagrams themselves</u> has been described. Instead, *Schuba* refers to a <u>half-open backlog queue</u>, as shown by the following excerpt from *Schuba*:

> In accordance with these classifications, monitor 52 sends a RST packet to unacceptable or bad addresses, closing corresponding connections to free resources of destination hosts 54. Further, an ACK packet is sent for suspect source addresses to free resources of the destination hosts 54 by <u>removing connections from ***a half-open backlog queue***</u>. Further, states 108 of state machine 100, conditionally coupled by transition paths 112, 118, 124, 126, 134, and 136, permit suspect address classification to change based on observed behavior of network traffic and asynchronous events, such as expiry and staleness event timers.

Schuba, col.11, ll.16-26 (emphasis added).

In contrast, claim 1 requires discarding the <u>datagram</u>, if the number of connectionless datagrams already queued to the port from the host exceeds the prescribed threshold. One of ordinary skill would instantly recognize the difference between discarding a datagram queued at a port and removing connections from a half-open backlog queue.

As an aid to understanding the distinction between this claimed feature and queues for half-open connections, the Applicants' Specification states:

> The first threshold is dynamically determined in the preferred embodiment. The owner of a server specifies for each port that is subject to datagram flooding checks a maximum number of queued datagrams (M) allowed at any time to the port and a controlling percentage (P) of available queue slots remaining for the port. The invention keeps track of the number (A) of queued datagrams for the port and it calculates the number of available queue slots (I) by subtracting the number of queued

> datagrams from the maximum number of datagrams (I = M – A). If the number of datagrams already queued for the transmitting host is equal to or greater than P times the number of queue slots left (=> P*I), then the present datagram is refused. Otherwise, the datagram is queued and the number of queued datagrams (A) for the port is incremented by one.

Specification, p.3, l.18 - p.4, l.6.

The Specification and the claims unambiguously state that the queue at issue is for datagrams. A half-open connection is not a datagram, even if a half-open connection is created using datagrams. *Schuba* only teaches methods for dealing with too many half-open connections, which is entirely distinct from discarding datagrams queued at a port. The fact that half-open connections are created with connectionless datagrams is wholly irrelevant to this distinction. Therefore, *Schuba* does not anticipate claim 1.

In the first paragraph of the Examiner's response, the Examiner states that "in viewing the arguments and lengthy case history with this application Examiner finds applicant is trying to differentiate the meaning of a connection attempt by using word such as connectionless or queuing the connectionless datagram." (Office Action dated May 2, 2006, p. 2.) However, the Examiner misconstrues Applicants' arguments. The thrust of Applicants' arguments is not directed towards splitting fine hairs over the meaning of the term "connectionless" or the meaning of the term "queuing the connectionless datagram." The thrust of Applicants' arguments is that a fundamental and marked difference exists between a queue of connectionless datagrams at a port, as claimed, and a queue of half-open connections, as described in *Schuba*. As shown above, the two features are entirely distinct. Anyone of ordinary skill in the art understands the distinctions between these two features, though the Examiner steadfastly refuses to recognize this fact in the face of clear disclosure in *Schuba* to the contrary. Additionally, the Examiner does not offer any support for the Examiner's assertions.

For example, in the second quoted paragraph, the Examiner states that, "The Office disagree [sic] with argument and notes that 'half open-connections' are interpreted to be equivalent to 'queuing the connectionless datagram'." (Office Action dated May 2, 2006, pp. 2-3.) The Examiner provides absolutely no support for the Examiner's assertions. The Examiner only quotes Applicants' argument and then asserts, without foundation or support, that "half open connections" are actually equivalent to "queuing the connectionless datagram." The Examiner provides no argument that the two are equivalent. Instead, the Examiner only misconstrues Applicants' argument and then concludes without any argument or support that the features are equivalent.

In view of the plain meaning of the claimed features and in view of the plain meaning of the teachings of *Schuba* quoted above, the Examiner's statements are incorrect. The Examiner failed to provide factual support for the assertion that a "half open connection" is equivalent to "queuing the connectionless datagram" and failed to submit an affidavit under 37 C.F.R. § 104(d)(2) attesting to the Examiner's personal knowledge in this area. Thus, the Examiner deprived the Applicants of an adequate opportunity to consider and refute the basis for the Examiner's assertions.

The rejection and the Examiner's arguments are manifestly incorrect in view of the plain meaning of the claims and of *Schuba*. *Schuba* does not teach all of the features of claim 1, as described above. Therefore, *Schuba* does not anticipate claim 1.

Nonetheless, the Examiner further asserts that:

> In response to applicant's argument on page 8, "*Schuba* only teaches methods for dealing with too many half-open connection, which entirely distinct from discarding datagrams queued at a port . . . The thrust of Applicants argument is not directed towards splitting fine hairs over the meaning of the term "connectionless" or the meaning of the term "queuing the connectionless datagram. The thrust of Applicants' arguments is that a fundamental and marked difference exist between a queue of connectionless datagrams at a port, as claimed, and a queue of half-open connections, as described in *Schuba*. The Office disagrees with argument to establish a connection the standard TCP/IP three-way handshake must occur, that is how a connection is established. TCP/IP transfers connectionless datagrams. Discarding datagrams queued at a port, when there are too many half open-connections is the same meaning. Note to queue a port is to start communication, which can be termed a half open-connection.

Final Office Action of October 20, 2006, pp. 3-4.

The Examiner asserts that discarding datagrams queued at a port is the same as discarding too many half open connections because a TCP/IP connection requires a three-way handshake and because TCP/IP transfers connectionless datagrams. However, the Examiner's response ignores the fact that a queue of half-open connections, as in *Schuba*, is still fundamentally different than a queue of connectionless datagrams, as in claim 1. The Examiner's assertion to the contrary is plainly wrong.

For example, one of ordinary skill knows that a half-open connection refers to a TCP connection that is partially open. The TCP protocol has a three state system for opening a connection. First, the originating site (A) sends a SYN packet to the destination (B). A is now half-open, and awaiting a response. B now updates its kernel information to indicate the incoming connection from A, and sends out a request to open a channel back (the SYN/ACK packet). At this point, B is now "half-open" (it has sufficient information to receive packets, but not enough to send packets back). Note that B was put into this state by another machine, outside of B's control.

Thus, once a computer receives a SYN packet, the computer is in a half-open state. In a half open state the computer updates its kernel information to indicate the incoming connection from a remote computer. The computer has sufficient information to receive, but not send packets. *However, the computer is not accumulating a queue of* ***datagrams***. Instead, the computer can, via the kernel, accumulate a number of these half-open states in which the computer is anticipating future datagrams. *Schuba* discusses discarding queues of these half-open connections. Thus, *Schuba* inherently teaches issuing a command to a kernel to discard half open connections. Claim 1 requires discarding connectionless datagrams, themselves, which is an entirely different thing. These two techniques are manifestly different from each other.

To further illustrate the distinction, *Schuba* can receive ten SYN datagrams and, as a result, establish ten half-open connections. If ten half-open connections exceeds a pre-defined limit of half-open connections, *Schuba* teaches discarding some of those ten half-open connections. The datagrams used to open the half-open connections are not discarded. In stark contrast, claim 1 requires discarding connectionless datagrams themselves. Thus, if a computer is receiving too many datagrams, then the datagrams are discarded, not the half-opened connections, as in *Schuba*.

Therefore, the Examiner's assertions that *Schuba* teaches features equivalent to those in claim 1 is manifestly incorrect. Accordingly, *Schuba* does not teach the features of claim 1. Therefore, *Schuba* does not anticipate claim 1 or any other claim in this grouping of claims.

B. GROUND OF REJECTION 2 (Claims 2, 4, 6, and 9-13)

The second ground of rejection is whether the Examiner failed to state a *prima facie* obviousness rejection against claims 2, 4, 6, and 9-13 under 35 U.S.C. §103(a) over *Schuba* in view of *Yavatkar et al.*, Method and System for Diagnosing Network Intrusion, U.S. Patent 6,735,702, August 31, 1999 (hereinafter "*Yavatkar*")

B.1. Claims 2, 4, and 6

Claim 2 is a representative claim in this grouping of claims. Claim 2 is as follows:

> 2. The method of claim 1 wherein the determining if the number of datagrams already queued to the port from the host exceeds a prescribed threshold further comprises:
>
> calculating the prescribed threshold by multiplying a percentage by the number of available queue slots for the port.

Regarding claim 2, the Examiner states that:

> As to dependent claim 2, the following is not taught in '378 "wherein the determining if the number of datagrams already queued to the port from the host exceeds a prescribed threshold further comprises: calculating the prescribed threshold by multiplying a percentage by the number of available queue slots for the port" however '702 teaches "A watchdog agent may assume a network attack exist if network congestion is detected . . . In an alternate embodiment a watchdog agent detects network congestion by monitoring interface discard counts and average queue lengths for each port on the node" in col. 15, line 63 through col. 16, line 17.
>
> It would have been obvious to one of ordinary skill in the **art** at the time of the invention to modify the teachings of '378 a method to protect a network from denial of service attacks to include a means to calculate the threshold limit per port. One of ordinary skill in the art would have been motivated to perform such a modification in order to gain information needed to diagnose a network attack (see '702 col. 2 lines 44 et seq.) "Therefore there exists a need for a system and method allowing for the distributed state of a network such as information about attack traffic, to be quickly and accurately collected. A system and method are needed for quickly and accurately diagnosing network attacks by determining information such as the source of, or a partial path of, attack traffic".

Final Office Action of October 20, 2006, pp. 6-7 (emphasis in original).

Applicants first respond to the rejection by showing that the proposed combination, when considered as a whole, does not teach or suggest all of the features of claim 2. Applicants will then show that no proper motivation, teaching, or suggestion exists to combine the references.

B.1.i. The Proposed Combination Does Not Teach or Suggest All of the Features of Claim 2

If the Patent Office does not produce a *prima facie* case of unpatentability, then without more Applicants are entitled to grant of a patent. *In re Oetiker*, 977 F.2d 1443, 1445, 24 U.S.P.Q.2d 1443, 1444 (Fed. Cir. 1992); *In re Grabiak*, 769 F.2d 729, 733, 226 U.S.P.Q. 870, 873 (Fed. Cir. 1985). A *prima facie* case of obviousness is established when the teachings of the prior art itself suggest the claimed subject matter to a person of ordinary skill in the art. *In re Bell,* 991 F.2d 781, 783, 26 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1993). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 32 U.S.P.Q.2d 1031, 1034 (Fed. Cir. 1994).

The Examiner failed to state a *prima facie* obviousness rejection against claim 2 because the proposed combination, when considered as a whole, does not teach or suggest all of the features of claim 2. As shown in Section A, *Schuba* does not teach all of the claimed features of claim 1, from which claim 2 depends. Additionally, *Schuba* does not teach, suggest, or give any incentive to make the needed changes to reach claim 1. Absent the Examiner pointing out some teaching or incentive to implement *Schuba* and the feature of connectionless datagrams already queued to the port as claimed in the determining, discarding, and queuing steps, one of ordinary skill in the art would not be led to modify *Schuba* to reach the present invention when the reference is examined as a whole.

Moreover, *Yavatkar* fails to cure the deficiencies of *Schuba* because *Yavatkar* does not teach of suggest any of the features of claim 1. The Examiner does not assert otherwise. Instead, *Yavatkar* teaches the use of agents to monitor, detect, and diagnose network conditions, such as a network attack. However, *Yavatkar* fails to teach or suggest any of the features of claim 1.

Because neither *Schuba* nor *Yavatkar* teach or suggest all of the features of claim 1, the proposed combination of *Schuba* and *Yavatkar*, when considered as a whole, does not teach or suggest all of the features of claim 2, which depends from claim 1. In addition, claim 2 claims other additional combinations of features not disclosed by either *Schuba* or *Yavatkar*.

Additionally, neither *Schuba* nor *Yavatkar* teach or suggest the feature of calculating the prescribed threshold by multiplying a percentage by the number of available queue slots for the port. The Examiner asserts otherwise, citing the following portion of *Yavatkar*:

> A watchdog agent may assume a network attack exists if network congestion is detected. To detect network congestion a watchdog agent monitors the number of packets which were to be received at the node on which the watchdog agent operates but which have been lost. The watchdog agent detects lost incoming packets by monitoring the TCP/IP stack, maintained in the operating system. If the number of incoming lost packets rises above a certain level the watchdog agent concludes a congestion condition exists. A watchdog agent uses its worksheet (which in turn uses an SNMP service or another type of service) to periodically

> access the OS to monitor this information. Alternately, a service may be provided to monitor this information directly. In an alternate embodiment a watchdog agent detects network congestion by monitoring interface discard counts and average queue lengths for each port on the node on which it executes. Packets may be dropped by a port if its buffers become full when the port receives packets at a rate which is faster than the rate it can send the packets to its host node--a discard count of such dropped packets may be maintained by a port. The average queue length of a buffer in a port is a measure of how full the buffer is.

Yavatkar, col.15, l.63 – col.16, l.17.

Neither the cited portion nor any other portion teaches or suggests the feature of calculating the prescribed threshold by multiplying a percentage by the number of available queue slots for the port. The cited portion discloses a watchdog agent that monitors a node for network attacks. The cited portion discloses two methods by which the watchdog agent may detect a network attack. However, neither method relates to the feature of calculating the prescribed threshold by multiplying a percentage by the number of available queue slots for the port because neither method relates to multiplying the number of available queue slots by a percentage.

On the other hand, claim 2 recites the feature of calculating the prescribed threshold by multiplying a percentage by the number of available queue slots for the port. However, neither method used by the watchdog agent disclosed in the cited portion involves multiplying <u>anything</u> by a percentage, let alone multiplying a percentage by the number of available queue slots for the port.

For example, *Yavatkar* states that "a watchdog agent detects network congestion by monitoring interface discard counts and average queue lengths for each port on the node on which it executes." (*Yavatkar*, col.16, ll. 9-11.) However, the discard count only measures the number of dropped packets that occur "when the port receives packets at a rate which is faster than the rate it can send the packets to its host node." (*Yavatkar*, col.16, ll. 13-14.) No multiplication operation is disclosed by the cited statements, let alone calculating a prescribed threshold by multiplying a percentage by the number of available queue slots for the port, as claimed. Furthermore, the average queue length, also used by the watchdog agent, "is a measure of how full the buffer is." (*Yavatkar*, col.16, ll. 15-16.) However, *Yavatkar* nowhere teaches or suggests that calculating the average queue length uses a percentage operand in a multiplication operation. In fact, the cited portion nowhere mentions the word "percentage," let alone teach or suggest using a percentage as an operand in a multiplication operation. Because *Yavatkar* fails to teach or suggest that the watchdog agent performs any multiplication operation on a percentage operand, *Yavatkar* fails to teach or suggest the feature of calculating the prescribed threshold by multiplying a percentage by the number of available queue slots for the port.

Schuba fails to cure the deficiencies of *Yavatkar*. Indeed, the Examiner admits that "the following is not taught in '378 'wherein the determining if the number of datagrams already queued to the port from the host exceeds a prescribed threshold further comprises: calculating the prescribed threshold by multiplying a percentage by the number of available queue slots for the port.'" (Final Office Action dated October 20, 2006, pp. 5-6.)

Therefore, neither *Schuba* nor *Yavatkar* teach or suggest the claimed feature of calculating the prescribed threshold by multiplying a percentage by the number of available queue slots for the port. Accordingly, the proposed combination of *Schuba* and *Yavatkar*, when considered as a whole, does not teach or suggest all of the features of claim 2. Therefore, under the standards of *In re Lowry* and *In re Grabiak*, the Examiner failed to state a *prima facie* obviousness rejection of claim 2 or any other claim in this grouping of claims.

***B.1.ii.* The Examiner Failed to State a Proper Motivation, Teaching, or Suggestion to Combine the References**

A *prima facie* obviousness rejection against claim 2 has not been made because no proper teaching or suggestion to combine the references has been stated. A *prima facie* case of obviousness is established when the teachings of the prior art itself suggest the claimed subject matter to a person of ordinary skill in the art. *In re Bell*, 991 F.2d 781, 783, 26 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1993). A proper *prima facie* case of obviousness cannot be established by combining the teachings of the prior art absent some teaching, incentive, or suggestion supporting the combination. *In re Napier*, 55 F.3d 610, 613, 34 U.S.P.Q.2d 1782, 1784 (Fed. Cir. 1995); *In re Bond*, 910 F.2d 831, 834, 15 U.S.P.Q.2d 1566, 1568 (Fed. Cir. 1990). No such teaching or suggestion is present in the cited references and the Examiner has not pointed out any proper teaching or suggestion that is based on the prior art.

In the case at hand, one of ordinary skill in the art would not be motivated to modify *Schuba* by the teaching of *Yavatkar* because *Schuba* already possesses the advantage proposed by the Examiner. Specifically, the Examiner states that:

> One of ordinary skill in the art would have been motivated to perform such a modification *in order to gain information needed to diagnose a network attack*

Final Office Action dated October 20, 2006, p. 7 (emphasis added).

However, the hypothetical advantage proposed by the Examiner, namely, "to gain information needed to diagnose a network attack," does not actually exist because *Schuba* already teaches a way to gain information to diagnose a network attack. Specifically, *Schuba* provides as follows:

> One form of the present invention includes a unique computer network monitoring technique. A further form of the present invention includes a unique defense for denial of service attacks.
>
> In another form of the present invention, network messages passing to one or more hosts from an untrusted network are actively monitored. Suspect messages are identified. The behavior of each suspect message is tracked in terms of a number of conditionally coupled states to determine whether any of the suspect messages present a security threat requiring action.

Schuba, col.1, ll.61-67.

The cited portion of *Schuba* discloses actively monitoring network messages between hosts, and identifies suspect messages; hence, information is gained. The cited portion also discloses "determin[ing] whether any of the suspect messages present a security threat requiring action." By determining if a security threat is present, *Schuba* diagnoses a network attack. Because *Schuba* identifies suspect messages to determine if a security threat is present, *Schuba* already possesses the proposed advantage of gaining information needed to diagnose a network attack. Because *Schuba* already possesses the advantage proposed by the Examiner, *Schuba* does not show a shortcoming or need that would motivate one of ordinary skill in the art to achieve the advantage proposed by the Examiner. Because no need for the combination exists, one of ordinary skill in the art would not be motivated to modify *Schuba* with the teaching of *Yavatkar*. Accordingly, the Examiner has failed to state a *prima facie* obviousness rejection against claim 2.

B.1.iii. The Proposed Combination Changes the Principle of Operation of Schuba

The Examiner failed to state a *prima facie* obviousness rejection because the proposed combination changes the principle of operation of the primary reference. In combining references to show the claimed feature, the proposed modification cannot change the principle of operation of a reference. See *In re Ratti*, 270 F.2d 810, 123 (CCPA 1959) and MPEP 2143.01. If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious. *Id.*

In the case at hand, the proposed combination of *Schuba* and *Yavatkar* changes the principle of operation of the prior art invention being modified. *Schuba* discloses a computer network monitoring technique utilizing a monitoring resource that contains a monitoring program. Specifically, *Schuba* provides as follows:

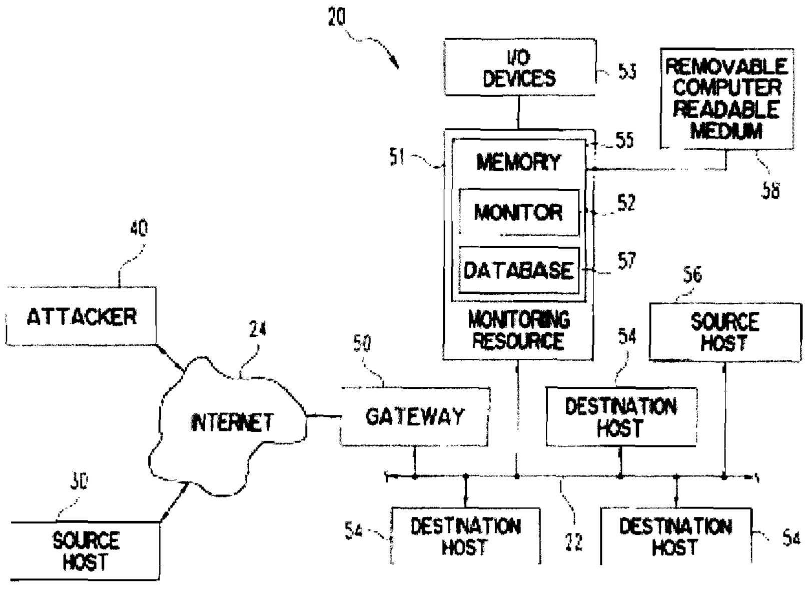

Fig. 3

Schuba, Figure 3.

> In addition to gateway 50, monitoring resource 51 is also operatively coupled to network 22. <u>A monitoring program, monitor 52, is configured for execution by monitoring resource 51</u> to protect other resources on network 22 from SYN flooding by sources from untrusted network 24, such as from attacker 40, and still facilitate establishment of connections with legitimate external sources, such as source host 30.

> Among the resources protected by monitor 52 are a number of destination hosts 54.
>
> ....
>
> <u>It is preferred that monitoring resource 51 be in the form of a programmable digital computer that is operable to execute monitor 52. Monitoring resource 51 may be in the form of a single processing unit operatively coupled to network 22 or a distributed system of different units operatively coupled to network 22, as would occur to those skilled in the art.</u> Resource 51 includes various hardware elements such as one or more Input/Output (I/O) devices 53 and memory 55. It is preferred I/O devices 53 include at least one means of administrative input to facilitate selective communication and control of monitor 52. I/O devices 53 may include a keyboard, mouse, visual display, and/or printer to name only a few possibilities.
>
> ....
>
> Monitor 52 is preferably arranged to capture IP/TCP datagrams passing along network 22 regardless of source or destination, including any TCP packets involved in a three-way handshake with destinations 54. Preferably, monitor 52 does not block or interfere with the transmission of packets to destination hosts 54, but rather examines the packets and reacts by selectively generating packets for one or more of destination hosts 54. Further, when a destination host 54 responds with a packet, such as a SYN+ACK packet, monitor 52 permits it to proceed. Monitor 52 is also arranged to asynchronously time selected events and respond to administrative inputs.

Schuba, col.6, l.54 – col.7, l.49 (emphasis added; omitted paragraphs indicated by ellipses).

The cited portion discloses that the monitoring resource 51 executes a monitor program 52, which performs various network monitoring functions, including examining packets. *Schuba* nowhere discloses that the monitoring program 52 may move from monitoring resource to monitoring resource, but instead discloses only that the monitoring program 52 performs monitoring functions from the monitoring resource 51 on which it resides. Thus, the principle of operation of *Schuba's* network protection system is to monitor a network using a monitoring program 52 that <u>fixedly resides</u> on one or more processing units, such as monitoring resource 51. *Schuba* does not teach any other mechanism for monitoring the network. Monitoring the network by any principle other than that described in *Schuba*, such as by deploying <u>mobile</u> software modules, would mean modifying, altering, or replacing the principle of operation of *Schuba's* invention.

The Examiner's proposed combination changes *Schuba's* principle of operation, namely, monitoring a network with a monitoring program that fixedly resides on one or more processing units, because *Yavatkar* teaches using mobile software modules that move from node to node within a network. Specifically, *Yavatkar* provides:

> The system and method of an exemplary embodiment of the present invention use agents--mobile software modules--to collect data on the state of a network during a network attack, allowing for more accurate diagnosis of an attack. During a network attack, the system and method of the present invention allow for details on the attack traffic (e.g., the source of the attack traffic and path of the attack traffic) to be gathered. The source of the attack traffic may be the originator of the attack traffic or, for example a gateway allowing attack traffic to enter a network and which is, in effect, the source of attack traffic to the network. Such information then may be used to halt the attack or insulate the network from the attack.
>
> When used herein, an agent is a software module having the capability to move from node to node on a network and to execute on the nodes to which it moves. In an exemplary embodiment of the present invention an agent may be, for example, an application functioning to detect or diagnose a network attack, but may also provide other functionality, such as altering a routing table or serving as a user application.

Yavatkar, col.3, ll.25-46.

Therefore, *Yavatkar* changes the principle of operation of *Schuba* by teaching the use of mobile software modules that move between nodes of a network, while *Schuba's* system relies on a monitoring program that fixedly resides on one or more processing units. Therefore, the teachings of *Schuba* and *Yavatkar* fails to render claim 2, or any other claim in this grouping of claims, *prima facie* obvious.

B.2. Claims 9 and 12

B.2.i. Response to Rejection

Claim 9 is a representative claim in this grouping of claims. Claim 9 is as follows:

> 9. The method of claim 1 further comprising:
> configuring a maximum number of connectionless datagrams allowed to be queued at the port.

Regarding claim 9, the Examiner states that:

> As to dependent claim 9, "further comprising: configuring a maximum number of connectionless, datagrams allowed to be queued at the port" is

> taught in '702 col. 12, lines 27-39 "In step 440, proactive environment 100 instantiates service object 300 based on the class of service 102. Proactive environment 100 configures service object 300 per the permissioning accessed in step 434. For example, one set of permissioning may allow agent 1 10 to use service object 300 to read the operating characteristics of port 21 and alter settings for the port".

Final Office Action dated October 20, 2006, pp. 7-8.

Applicants first respond to the rejection by showing that the proposed combination, when considered as a whole, does not teach or suggest all of the features of claim 9. Applicants will then show that the no proper motivation, teaching, or suggestion exists to combine the references.

B.2.i.a. The Proposed Combination Does Not Teach or Suggest All of the Features of Claim 9

The Examiner failed to state a *prima facie* obviousness rejection against claim 9 because the proposed combination, when considered as a whole, does not teach or suggest all of the features of claim 9. As shown in Section A, *Schuba* does not teach all of the claimed features of claim 1, from which claim 9 depends. Additionally, *Schuba* does not teach, suggest, or give any incentive to make the needed changes to reach claim 1. Absent the Examiner pointing out some teaching or incentive to implement *Schuba* and the feature of connectionless datagrams already queued to the port as claimed in the determining, discarding, and queuing steps, one of ordinary skill in the art would not be led to modify *Schuba* to reach the present invention when the reference is examined as a whole.

Moreover, *Yavatkar* fails to cure the deficiencies of *Schuba* because *Yavatkar* does not teach or suggest any of the features of claim 1. The Examiner does not assert otherwise. Instead, *Yavatkar* teaches the use of agents to monitor, detect, and diagnose network conditions, such as a network attack. However, *Yavatkar* fails to teach or suggest any of the features of claim 1.

Because neither *Schuba* nor *Yavatkar* teach or suggest all of the features of claim 1, the proposed combination of *Schuba* and *Yavatkar*, when considered as a whole, does not teach or suggest all of the features of claim 9, which depends from claim 1. In addition, claim 9 claims other additional combinations of features not disclosed by either *Schuba* or *Yavatkar*.

Neither *Schuba* nor *Yavatkar* teach or suggest the feature of configuring a maximum number of connectionless datagrams allowed to be queued at the port. The Examiner asserts otherwise, citing the following portion of *Yavatkar*:

> In step 440, proactive environment 100 instantiates service object 300 based on the class of service 102. Proactive environment 100 configures service object 300 per the permissioning accessed in step 434. For example, one set of permissioning may allow agent 110 to use service object 300 to read the operating characteristics of port 21 and alter settings for the port, and another set of permissioning may allow agent 110 to use service object 300 only to read the operating characteristics of port 21. Proactive environment 100 sets permission variables 312, members of service object 300, to indicate which aspects of service 102 (in the form of methods 322-326 of service object 300) agent 110 may access.

Yavatkar, col.12, ll.27-39.

Neither the cited portion nor any other portion of *Yavatkar* teaches or suggests the feature of configuring a maximum number of connectionless datagrams allowed to be queued at the port. The cited portion describes one phase of the interaction between an agent and a service. Specifically, the cited portion discloses instantiating a service object based on an agent's access control list so that the agent can read the operating characteristics for a particular port. However, the cited portion nowhere discloses any threshold levels, such as a maximum, as claimed in claim 9.

On the other hand, claim 9 recites the feature of configuring a maximum number of connectionless datagrams allowed to be queued at the port. As a first matter, the cited portion nowhere teaches or suggests configuring a maximum number of anything. For example, the cited portion states that "[p]roactive environment 100 configures service object 300 per the permissioning accessed in step 434." However, the cited portion discloses only configuring a "service object," but nowhere teaches or suggests configuring a maximum number of anything, let alone configuring a maximum number of connectionless datagrams allowed to be queued at the port. In fact, the cited portion discloses no threshold levels at all, such as maximum or a minimum, and therefore cannot teach or suggest configuring a maximum number of connectionless datagrams allowed to be queued at the port, as claimed.

As a second matter, the cited portion nowhere teaches or suggests any "connectionless datagrams allowed to be queued at the port," as claimed, because the cited portion does not relate to any queuing operations at all. Instead, the cited portion discloses only the interaction between an agent and a service once access has been granted. Because the cited portion fails to teach or suggest, and does not relate to, any queues at all, the cited portion fails to teach or suggest the feature of configuring a maximum number of connectionless datagrams allowed to be queued at the port.

Schuba fails to cure the deficiencies of *Yavatkar*. *Schuba* discloses a monitoring system to prevent denial of service attacks on a network. However, *Schuba* fails to disclose any of the features of claim 9. The Examiner tacitly admits that *Schuba* teaches or suggests none of the features of claim 9 by citing *Yavatkar* with respect to the features of claim 9.

Therefore, neither *Schuba* nor *Yavatkar* teach or suggest the claimed feature of configuring a maximum number of connectionless datagrams allowed to be queued at the port. Accordingly, the proposed combination of *Schuba* and *Yavatkar*, when considered as a whole, does not teach or suggest all of the features of claim 9. Therefore, under the standards of *In re Lowry* and *In re Grabiak*, the Examiner failed to state a *prima facie* obviousness rejection of claim 9 or any other claim in this grouping of claims.

B.2.i.b. No Proper Teaching, Motivation, or Suggestion Exists to Combine the References as Proposed by the Examiner

Because the Examiner attempts to combine *Schuba* and *Yavatkar* to reject claim 9 as obvious, and because the Examiner presumably asserts the same rationale for combining *Schuba* and *Yavatkar* with respect to claim 9, the same analysis applied in Section B.1.ii. and B.1.iii. regarding the improper combination of *Schuba* and *Yavatkar* also applies to claim 9. Specifically, *Schuba* and *Yavatkar* may not be combined because the Examiner fails to state a proper motivation, teaching, or suggestion to combine the references and because the proposed combination changes the principle of operation of *Schuba*. Accordingly, no *prima facie* obviousness rejection can be made against claim 9 or any other claim in this grouping of claims.

B.2.ii. Rebuttal to Examiner's Response

In response to the facts established above, the Examiner asserts that:

> In response to applicant's argument beginning on page 11, "In addition, the proposed combination does not teach all of the features of the other dependent claims . . . the cited text plainly does not teach or suggest configuring a maximum number of connectionless datagrams allowed to be queued at the port". The Examiner disagrees and notes that the setting for the port that *Yavatkar* can alter are obviously the maximum number of connections allowed for the port. The references should be looked at in

combination *Schuba* teaches that a limit is set by the TCP/IP protocol for the maximum number of connections allowed to be established to a port, *Yavatkar* teaches that the operational settings of a port can be altered, these operational settings are an obvious variation of the number of connections allowed at a port.

Final Office Action dated October 20, 2006, p. 4.

The Examiner states that "[t]he references should be looked at in combination *Schuba* teaches that a limit is set by the TCP/IP protocol for the maximum number of connections allowed to be established to a port." (Final Office Action dated October 20, 2006, p. 4.) However, *Schuba's* disclosure regarding the limit on the number of concurrent half-open connections at a port does not relate to *Yavatkar's* agent/port interaction. *Schuba* discloses only that "[t]here is a limit on the number of concurrent TCP connections that can be in a half-open connection state." (*Schuba*, col. 4, ll. 54-55.) However, *Schuba* fails to teach or suggest altering the maximum number of half-open connections, let alone altering the maximum number of half-open connections in the context of mobile agent/port interactions. Because *Schuba* does not teach or suggest that the maximum number of half-open connections may be altered, and because *Schuba* nowhere teaches or suggests the use of mobile agents to alter port settings, one of the ordinary skill in the art would not glean from *Schuba* that *Yavatkar's* agent/port interaction to alter port settings would include configuring a maximum number of connections allowed at a port, as claimed.

Nonetheless, the Examiner states that "*Yavatkar* teaches that the operational settings of a port can be altered, these operational settings are an obvious variation of the number of connections allowed at a port." (Final Office Action dated October 20, 2006, p. 4.) However, the Examiner has stated an illogical relationship between "operational settings" and "the number of connections allowed at a port" by stating the former may be an "obvious variation" of the latter. The alteration of operational settings, as described in *Yavatkar*, involves an agent/port interaction that covers a wide range of parameters. On the other hand, the number of connections allowed at a port is a single parameter. Hence, the Examiner's statement equates to stating that food is an obvious variation of an apple. Instead, the number of connections allowed at a port can potentially be a <u>type</u> of operational setting, just as an apple is a type of food.

However, *Yavatkar* nowhere teaches or suggests that the number of connections allowed at a port relates to the operational settings that may be altered by the agent/port interaction. For example, *Yavatkar* states that "one set of permissioning may allow agent 110 to use service object

300 to read the operating characteristics of port 21 and alter settings for the port." (*Yavatkar*, col. 12, ll. 30-33.) However, the cited statement discloses only that an agent may alter setting for a port, but discloses nothing about the nature of those settings, let alone that altering port settings relates to configuring a <u>maximum</u> number of connectionless datagrams allowed to be queued at the port. Hence, because neither *Schuba* nor *Yavarkar* relates or connects operational settings to the number of connections allowed at a port, and because an operational setting cannot logically be an obvious variation of the number of connections allowed at a port, neither reference teaches or suggests that operational settings are an obvious variation of the number of connections allowed at a port.

Therefore, neither *Schuba* nor *Yavatkar* teach or suggest the claimed feature of configuring a maximum number of connectionless datagrams allowed to be queued at the port. Accordingly, the proposed combination of *Schuba* and *Yavatkar*, when considered as a whole, does not teach or suggest all of the features of claim 9. Therefore, under the standards of *In re Lowry* and *In re Grabiak*, the Examiner failed to state a *prima facie* obviousness rejection of claim 9 or any other claim in this grouping of claims.

B.3. Claims 10 and 13

Claim 10 is a representative claim in this grouping of claims. Claim 10 is as follows:

> 10. The method of claim 9 wherein the configuring step further includes configuring a controlling percentage of available queue slots remaining for the port; and wherein the prescribed threshold is based on the controlling percentage of available queue slots remaining for the port.

Regarding claim 10, the Examiner states:

> As to dependent claim 10, "wherein the configuring step further includes configuring a controlling percentage of available queue slots remaining for the port; and wherein the proscribed threshold is based on the controlling percentage of available queue slots remaining for the port" is shown in '702 col. 12, lines 27-39.

Final Office Action dated October 20, 2006, p. 8.

Applicants first respond to the rejection by showing that the proposed combination, when considered as a whole, does not teach or suggest all of the features of claim 10. Applicants will then show that the no proper motivation, teaching, or suggestion exists to combine the references.

B.3.i. The Proposed Combination Does Not Teach or Suggest All of the Features of Claim 10

The Examiner failed to state a *prima facie* obviousness rejection against claim 10 because the proposed combination, when considered as a whole, does not teach or suggest all of the features of claim 10. As shown in Section B.2.i.a., *Schuba* does not teach or suggest all of the claimed features of claim 9, from which claim 10 depends. Moreover, *Yavatkar* fails to cure the deficiencies of *Schuba* because *Yavatkar* does not teach or suggest any of the features of claim 9.

Because neither *Schuba* nor *Yavatkar* teach or suggest all of the features of claim 9, the proposed combination of *Schuba* and *Yavatkar*, when considered as a whole, does not teach or suggest all of the features of claim 10, which depends from claim 9. In addition, claim 10 claims other additional combinations of features not disclosed by either *Schuba* or *Yavatkar*.

Neither *Schuba* nor *Yavatkar* teach or suggest the feature of configuring a controlling percentage of available queue slots remaining for the port. The Examiner asserts otherwise, citing the following portion of *Yavatkar*:

> In step 440, proactive environment 100 instantiates service object 300 based on the class of service 102. Proactive environment 100 configures service object 300 per the permissioning accessed in step 434. For example, one set of permissioning may allow agent 110 to use service object 300 to read the operating characteristics of port 21 and alter settings for the port, and another set of permissioning may allow agent 110 to use service object 300 only to read the operating characteristics of port 21. Proactive environment 100 sets permission variables 312, members of service object 300, to indicate which aspects of service 102 (in the form of methods 322-326 of service object 300) agent 110 may access.

Yavatkar, col.12, ll.27-39.

Neither the cited portion nor any other portion of *Yavatkar* teaches or suggests the feature of configuring a controlling percentage of available queue slots remaining for the port. The cited portion describes one phase of the interaction between an agent and a service. Specifically, the cited portion discloses instantiating a service object based on an agent's access control list so that the agent can read the operating characteristics for a particular port. However, the cited portion teaches or suggests nothing relating to available queue slots for a port.

On the other hand, claim 10 recites the feature of configuring a controlling percentage of available queue slots remaining for the port. The cited portion of *Yavatkar* differs from the claimed feature because the cited portion does not relate to available queue slots for a port, let alone teach or suggest configuring a controlling percentage of available queue slots remaining for the port. For example, the cited portion states that "one set of permissioning may allow agent 110 to use service object 300 to read the operating characteristics of port 21 and alter settings for the port, and another set of permissioning may allow agent 110 to use service object 300 only to read the operating characteristics of port 21." However, the cited portion discloses only that an agent may "read the operating characteristics" or "alter settings" for port 21, but nowhere discloses anything relating to a controlling percentage of available queue slots remaining for the port. Because the cited portion does not relate to the feature of configuring a controlling percentage of available queue slots remaining for the port, *Yavatkar* fails to teach or suggest the claimed feature.

Schuba fails to cure the deficiencies of *Yavatkar*. *Schuba* discloses a monitoring system to prevent denial of service attacks on a network. However, *Schuba* fails to disclose any of the features of claim 10. The Examiner tacitly admits that *Schuba* teaches or suggests none of the features of claim 10 by citing *Yavatkar* with respect to the features of claim 10.

Therefore, neither *Schuba* nor *Yavatkar* teach or suggest the claimed feature of configuring a controlling percentage of available queue slots remaining for the port. Accordingly, the proposed combination of *Schuba* and *Yavatkar*, when considered as a whole, does not teach or suggest all of the features of claim 10. Therefore, under the standards of *In re Lowry* and *In re Grabiak*, the Examiner failed to state a *prima facie* obviousness rejection of claim 10 or any other claim in this grouping of claims.

B.3.ii. No Proper Teaching, Motivation, or Suggestion Exists to Combine the References as Proposed by the Examiner

Because the Examiner attempts to combine *Schuba* and *Yavatkar* to reject claim 10 as obvious, and because the Examiner presumably asserts the same rationale for combining *Schuba* and *Yavatkar* with respect to claim 10, the same analysis applied in Section B.1.ii. and B.1.iii. regarding the improper combination of *Schuba* and *Yavatkar* also applies to claim 10. Specifically, *Schuba* and *Yavatkar* may not be combined because the Examiner fails to state a proper motivation, teaching, or suggestion to combine the references and because the proposed combination changes the principle of operation of *Schuba*. Accordingly, no *prima facie* obviousness rejection can be made against claim 10 or any other claim in this grouping of claims.

B.4. Claim 11

Claim 11 is the only claim in this grouping of claims. Claim 11 is as follows:

> 11. The method of claim 1 wherein the port comprises a plurality of queue slots, the method further comprising:
>
> maintaining a number of available queue slots of the plurality of queue slots for the port.

Regarding claim 11, the Examiner states:

> As to dependent claim 11, "wherein the port comprises a plurality of queue slots the method further comprising: maintaining a number of available queue slots of the plurality of queue slots for the port" is disclosed in '702 col. 12, lines 27-39.

Final Office Action dated October 20, 2006, p. 8.

Applicants first respond to the rejection by showing that the proposed combination, when considered as a whole, does not teach or suggest all of the features of claim 11. Applicants will then show that the no proper motivation, teaching, or suggestion exists to combine the references.

B.4.i. The Proposed Combination Does Not Teach or Suggest All of the Features of Claim 11

The Examiner failed to state a *prima facie* obviousness rejection against claim 11 because the proposed combination, when considered as a whole, does not teach or suggest all of the features of claim 11. As shown in Section A, *Schuba* does not teach all of the claimed features of claim 1, from which claim 11 depends. Additionally, *Schuba* does not teach, suggest, or give any incentive to make the needed changes to reach claim 1. Absent the Examiner pointing out some teaching or incentive to implement *Schuba* and the feature of connectionless datagrams already queued to the port as claimed in the determining, discarding, and queuing steps, one of ordinary skill in the art would not be led to modify *Schuba* to reach the present invention when the reference is examined as a whole.

Moreover, *Yavatkar* fails to cure the deficiencies of *Schuba* because *Yavatkar* does not teach or suggest any of the features of claim 1. The Examiner does not assert otherwise. Instead, *Yavatkar* teaches the use of agents to monitor, detect, and diagnose network conditions, such as a network attack. However, *Yavatkar* fails to teach or suggest any of the features of claim 1.

Because neither *Schuba* nor *Yavatkar* teach or suggest all of the features of claim 1, the proposed combination of *Schuba* and *Yavatkar*, when considered as a whole, does not teach or suggest all of the features of claim 11, which depends from claim 1.

B.4.ii. No Proper Teaching, Motivation, or Suggestion Exists to Combine the References as Proposed by the Examiner

Because the Examiner attempts to combine *Schuba* and *Yavatkar* to reject claim 11 as obvious, and because the Examiner presumably asserts the same rationale for combining *Schuba* and *Yavatkar* with respect to claim 11, the same analysis applied in Section B.1.ii. and B.1.iii. regarding the improper combination of *Schuba* and *Yavatkar* also applies to claim 11. Specifically, *Schuba* and *Yavatkar* may not be combined because the Examiner fails to state a proper motivation, teaching, or suggestion to combine the references and because the proposed combination changes the principle of operation of *Schuba*. Accordingly, no *prima facie* obviousness rejection can be made against claim 11.

C. CONCLUSION

As shown above, *Schuba* does not anticipate the claims and the Examiner failed to state a *prima facie* obviousness rejection against any of the claims. Therefore, Applicants request that the Board of Patent Appeals and Interferences reverse the rejections. Additionally, Applicants request that the Board direct the Examiner to allow the claims.

/Theodore D. Fay III/
Theodore D. Fay III
Reg. No. 48,504
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

TF/ka

CLAIMS APPENDIX

The text of the claims involved in the appeal is as follows:

1. A method of preventing a flooding attack on a network server in which a large number of connectionless datagrams are received for queuing to a port on the network server, comprising:

determining, in response to the arrival of a connectionless datagram from a host for a port on the network server, if the number of connectionless datagrams already queued to the port from the host exceeds a prescribed threshold;

discarding the datagram, if the number of connectionless datagram already queued to the port from the host exceeds the prescribed threshold; and

queuing the connectionless datagram to a queue slot of the port, if the number of connectionless datagram already queued to the port from the host does not exceed the prescribed threshold.

2. The method of claim 1 wherein the determining if the number of datagrams already queued to the port from the host exceeds a prescribed threshold further comprises:

calculating the prescribed threshold by multiplying a percentage by the number of available queue slots for the port.

3. An apparatus for preventing a flooding attack on a network server in which a large number of datagrams are received for queuing to a port on the server, comprising:

means for determining, in response to a datagram from a host for the port on the network server, if the number of datagrams queued on the port by the host exceeds a prescribed threshold;

means for discarding the datagram, if the number of datagrams queued on the port by the host exceeds the prescribed threshold; and

means for queuing the datagram to a queue slot of the port, if the number of datagrams queued on the port by the host does not exceed the prescribed threshold.

4. The apparatus of claim 3 wherein the means for determining if the number of datagrams already queued to the port from the host exceeds a prescribed threshold further comprises:

means for calculating the prescribed threshold by multiplying a percentage by a number of available queue slots for the port.

5. A storage media containing program code that is operable by a computer for preventing a flooding attack on a network server in which a large number of datagrams are received for queuing to a port on the network server, the program code including instructions for causing the computer to execute the steps of:

determining, in response to receiving a datagram from a host for the port on the network server, if the number of datagrams already queued to the port from the host exceeds a prescribed threshold;

discarding the datagram, if the number of datagrams already queued to the port from the host exceeds the prescribed threshold; and

queuing the datagram to a queue slot of the port, if the number of datagrams already queued to the port from the host does not exceed the prescribed threshold.

6. The storage media of claim 5 wherein the program code includes further instructions for causing the computer to execute the step of:

calculating the prescribed threshold by multiplying a percentage by a number of available queue slots for the port.

9. The method of claim 1 further comprising:

configuring a maximum number of connectionless datagrams allowed to be queued at the port.

10. The method of claim 9 wherein the configuring step further includes configuring a controlling percentage of available queue slots remaining for the port; and wherein the prescribed threshold is based on the controlling percentage of available queue slots remaining for the port.

11. The method of claim 1 wherein the port comprises a plurality of queue slots, the method further comprising:

maintaining a number of available queue slots of the plurality of queue slots for the port.

12. The apparatus of claim 3 further comprising:

a means for configuring a maximum number of connectionless datagrams allowed to be queued at the port.

13. The apparatus of claim 12 wherein the means for configuring further comprises means for configuring a controlling percentage of available queue slots remaining for the port.

14. The storage media of claim 5 wherein the computer is the network server.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.